IN THE UNITED STATES PATENT AND TRADEMARK OFFICE

Application Serial No. 09/721,402
Confirmation No. 3481
Filing Date ..... November 22, 2000
Applicant ..... Evans et al.
Group Art Unit 2616
Examiner ..... Vincent F. Boccio
Attorney's Docket No. MS1-0688US
Title: IMPROVED DVD NAVIGATOR AND APPLICATION PROGRAMMING INTERFACES (APIS)

DECLARATION UNDER 37 C.F.R. § 1.131

As a below named inventor, I hereby declare that:

My residence, post office address and citizenship are as stated below next to my name.

I believe I am the original, first and joint inventor of the subject matter which is claimed and for which a patent is sought on the invention entitled "Improved DVD Navigator and Application Programming Interfaces (APIs)," as identified above.

The invention was conceived and reduced to practice in the United States at least as early as November 24, 1999, as evidenced by the following documents, copies of which are attached:

(1) a redacted copy of an Invention Disclosure document giving rise to the subject application, which establishes conception and reduction to practice of the claimed invention at least as early as the Beta 2 release of Windows® Millennium operating system (the Beta 2 release was a limited release under a non-disclosure agreement),

(2) a publication entitled "Frequently Asked Questions – Windows Millennium Edition," available on the ACTIVENETWORK website at www.activewin.com/faq/millennium.shtml, indicating that Windows® Millennium

Beta 2 edition was released on November 23, 1999 (see Timeline at end of publication), and

(3) a publication entitled "Introducing Windows Millennium Edition ('Windows Me') Beta 3," available on Paul Thurrott's SuperSite for Windows at www.winsupersite.com/reviews/millennium_b3_intro.asp, indicating that "On November 24, 1999, Microsoft released Windows Millennium Beta 2 ..." (see paragraph 6).

The dates and other proprietary information listed on the Invention Disclosure document have been redacted, as permitted by MPEP § 715.07(II). However, Applicants submit that each of the features of the pending claims was conceived of and reduced to practice at least by the release of the Beta 2 release of Windows® Millennium operating system, which occurred on or about November 24, 1999, as evidenced by the foregoing publications. As noted above, the Beta 2 release was a limited release made subject to a non-disclosure agreement.

All statements made herein of my own knowledge are true, and all statements made on information and belief are believed to be true. Further, these statements are made with the knowledge that willful false statements and the like so made are punishable by fine or imprisonment, or both, under Section 1001 of Title 18 of the United States Code and that such willful false statement may jeopardize the validity of the application or any patent issued therefrom.

Full name of inventor: Glenn F. Evans

Inventor's Signature: Glenn Evans Date: Jan 12, 2006

Residence: Kirkland, WA

Post Office Address: One Microsoft Way, Redmond, WA 98052

Citizenship: Canada

Full name of inventor: Alok Chakrabarti

Inventor's Signature: _____ Date: _____

Residence: Bellevue, WA

Post Office Address: One Microsoft Way, Redmond, WA 98052

Citizenship: India

IN THE UNITED STATES PATENT AND TRADEMARK OFFICE

Application Serial No. 09/721,402
Confirmation No. 3481
Filing Date ..... November 22, 2000
Applicant ..... Evans et al.
Group Art Unit 2616
Examiner ..... Vincent F. Boccio
Attorney's Docket No. MS1-0688US
Title: IMPROVED DVD NAVIGATOR AND APPLICATION PROGRAMMING INTERFACES (APIS)

DECLARATION UNDER 37 C.F.R. § 1.131

As a below named inventor, I hereby declare that:

My residence, post office address and citizenship are as stated below next to my name.

I believe I am the original, first and joint inventor of the subject matter which is claimed and for which a patent is sought on the invention entitled "Improved DVD Navigator and Application Programming Interfaces (APIs)," as identified above.

The invention was conceived and reduced to practice in the United States at least as early as November 24, 1999, as evidenced by the following documents, copies of which are attached:

(1) a redacted copy of an Invention Disclosure document giving rise to the subject application, which establishes conception and reduction to practice of the claimed invention at least as early as the Beta 2 release of Windows® Millennium operating system (the Beta 2 release was a limited release under a non-disclosure agreement),

(2) a publication entitled "Frequently Asked Questions – Windows Millennium Edition," available on the ACTIVENETWORK website at www.activewin.com/faq/millennium.shtml, indicating that Windows® Millennium

Beta 2 edition was released on November 23, 1999 (see Timeline at end of publication), and

(3) a publication entitled "Introducing Windows Millennium Edition ('Windows Me') Beta 3," available on Paul Thurrott's SuperSite for Windows at www.winsupersite.com/reviews/millennium_b3_intro.asp, indicating that "On November 24, 1999, Microsoft released Windows Millennium Beta 2 ..." (see paragraph 6).

The dates and other proprietary information listed on the Invention Disclosure document have been redacted, as permitted by MPEP § 715.07(II). However, Applicants submit that each of the features of the pending claims was conceived of and reduced to practice at least by the release of the Beta 2 release of Windows® Millennium operating system, which occurred on or about November 24, 1999, as evidenced by the foregoing publications. As noted above, the Beta 2 release was a limited release made subject to a non-disclosure agreement.

All statements made herein of my own knowledge are true, and all statements made on information and belief are believed to be true. Further, these statements are made with the knowledge that willful false statements and the like so made are punishable by fine or imprisonment, or both, under Section 1001 of Title 18 of the United States Code and that such willful false statement may jeopardize the validity of the application or any patent issued therefrom.

Full name of inventor: Glenn F. Evans

Inventor's Signature: _____ Date: _____

Residence: Kirkland, WA

Post Office Address: One Microsoft Way, Redmond, WA 98052

Citizenship: Canada

Full name of inventor: Alok Chakrabarti

Inventor's Signature:  Date: 01.19.2006.

Residence: Bellevue, WA

Post Office Address: One Microsoft Way, Redmond, WA 98052

Citizenship: India

DVD2 API Patent Disclosure

Prior Disclosures

This change to the DVD APIs was implemented in the Microsoft DVD Navigator component, which was included in the Beta 2 release of Windows Millennium

Introduction

A DVD player is composed of three logical units (as defined in the DVD specification):

- the DVD player application that presents an interface to the user and relays user commands to the DVD Navigator
- the DVD Navigator that reads and interprets the information on the DVD disc and controls which segments of video and audio are processed based on the user commands
- the DVD presentation layer which decompresses and presents the audio, video and subpicture streams based on what the Navigator presents to it

DVD application programs control the DirectShow DVD Navigator filter through a specialized Application Programming Interface (API) known as the Microsoft DVD2 API. The DVD2 API enables an application to interactively control the playback of DVD content. The API consists of two interfaces -- IDvdInfo2 and IDvdControl2. An application uses the IDvdInfo2 interface to query the current state of the Navigator and uses the IDvdControl2 interface to control playback or to alter the Navigator's state.

The IDvd2 API has several unique, novel features:

- Thread synchronization methods for real-time playback
- Playback control mechanism to determine the degree of interactivity
- Application program to disc program communication mechanisms
- Simplified naming scheme
- Playing of time ranges (PlayPeriodInTitle)
- Mechanism for coordinating handling parental level requests
- Mechanism for determining the minimal parental level to play a restricted segment of content
- UniqueID algorithm to uniquely identify DVD discs.
- Bookmarking of any location within content

Problems Addressed

Implementing a DVD navigator is a complex task. An application that wants to integrate DVD video, must implement a DVD Navigator component. Microsoft provides a standard, specification-compliant DVD Navigator as part of Windows to help application developers avoid this repetitive and difficult task. The DVD2 API provides a simplified, consistent interface that applications can use to control a standard DVD Navigator. The API interface influences the flexibility and usefulness of the underlying DVD Navigator. The DVD2 API addresses several issues:

1. The DVD2 API adds synchronization mechanisms between the application and the DVD Navigator. Since the DVD playback runs concurrently with the player application, the application must either

wait for each command to be executed or create a separate thread (execution unit/path?) to asynchronously handle the command. If it waits for each command to complete, the program could become quite unresponsive. Alternatively, the application becomes a lot more complex, if it has to manage a separate thread to handle the requests and take appropriate action at the correct point in time. The DVD2 API allows an application to either wait for completion or be notified of the completion (with execution result) by using a few flags and the built-in features of the API.

2. Current DVD APIs do not provide any control over the balance between being interactive and producing seamless playback. A user may want to see changes immediately when they click a button and disregard any video that they were watching. However, an automated player might want to finish playing a portion of content before switching to another part. The DVD2 API allows choice of either mode via some pre-defined flags specified by the application, which was not possible before.
3. The DVD Navigator simulates a virtual DVD CPU with state information that executes programs contained on every DVD disc. DVD APIs do not provide any means for application programs to communicate with this virtual CPU. Custom applications can read information from the disc's program and control its playback. A communications mechanism has now been enabled to implement PC-specific special features such as "Controlled Unlocking" (allowing a user to view restricted portions of the disc after obtaining a key or password) as part of enhanced DVDs.
4. The naming scheme proposed in the DVD specification (Annex J) is often confusing to developers who are not intimately familiar with DVD jargon and terminology. Also many command(?) names do not accurately describe their function. The DVD2 API release makes the naming scheme a natural one, where the name suggests what the command should be do.
5. The new API has an in-built mechanism for give a more precise playback of sections of content.
6. The DVD specification provides two default mechanisms to enforce parental controlled viewing:
    a. The playback path can branch into two paths at a point in the content (the 'minimum parental level branching'). The failure branch is taken if the current parental level is too low and the player application decides that the user cannot change the level. Otherwise, the user is allowed to change to the required parental level (e.g., via password) to allow going on the success branch. The DVD2 API coordinates with the player application to correctly handle parental level changes. The Navigator queries the application to decide if the application will allow the Navigator to increase the parental level (based on user action).
    b. The video can branch into different segments based on the actual value of the parental level ('multisegment parental level branching'). If no segment is available for the current parental level then the playback stops.

- The DVD2 API simplifies application authoring, adds functionality and solves many difficult synchronization issues common to DVD player applications dvelopment.
- A common DVD API helps discourage proprietary single-use monolithic DVD solutions that serve only as standalone DVD players. It also allows various applications (such as presentation programs, DVD players, games, or interactive learning programs) to add DVD support without having to know which DVD decoder or DVD hardware support is on the system. Historically, custom DVD solutions tend to be very Operating System and hardware dependent and have limited upgrade options for users.

The Invention Vs. Previous Approaches

- The DVD2 API adds flexible synchronization mechanisms for the application to know the completion status of requests made to the DVD Navigator. The new command completion notification allows the application to concurrently perform other tasks and be informed of the status of a previous request. Other DVD APIs assumed that either the application would be blocked until the request was completed, or would not send any notification to the application. Applications now have the option of receiving a synchronization object that they can use to wait on or are notified about completion events.
- The synchronization mechanism also returns the status of the request that indicates whether it succeeded or returns the reason (an error code) for it's the failure.

The new mechanism also notifies the application of every request that is cancelled or overridden by the disc's actions or by further user actions.

- Current DVD APIs use predefined behaviors that dictate how a command interacts with the current display. When an application issues a new request, it pre-empts and cancels any content (video or audio) that is being played. Alternatively, the APIs semantics dictate that the current presentation completes before the new content is presented which forces the user to wait before he/she can request another action. Interactive applications such as DVD players and games may require the first behavior (instant effect), but other applications such as a slideshow may require the second behavior (complete the current presentation). Since these two options are mutually exclusive, predefined APIs cannot accommodate both. The DVD2 API allows the application to indicate the desired behavior via some flags, and also how it interacts with the synchronization mechanism,
- The DVD Navigator simulates a virtual CPU that uses an execution state (in the form of a set of memory registers). The first MicroSoft DVD API allowed applications to read the contents of the registers. The DVD2 API also allows the application program to also change the contents of the memory registers. The combined read/write functionality allows DVD applications to 'communicate' with the program on the disc. The read and write methods works in such a way that they can also be used for synchronization. With read/write functionality, an application can implement 'controlled unlocking' or restricted access to the DVD content. With controlled unlocking, the user is restricted from viewing portions of the disc until the player application sets specific memory registers. The player application could receive this information from the content's author or from a website.

- The DVD2 API contains a simplified naming scheme for the potential user operations suggested in the DVD specification Annex J. The DVD2 API uses less DVD jargon and features a more intuitive naming scheme. The user operation names proposed in the DVD specification are unclear and can lead to incorrect usage or under-utilization by application programs. The names now suggest their usage instead of an abstract label. Also time codes are now returned in a simple integer format instead of the awkward BCD coding.
- Parental level management

The DVD2 API has a mode that pauses the Navigator and lets the application respond to the parental level increase request before the Navigator continues. If the increase request is granted, the playback continues without requiring the user to start the movie from the beginning.

- The DVD specification only states that the Navigator should pause until it knows whether the request succeeded or failed. It does not describe a mechanism to accomplish this task and suggests that the Navigator "calls the Temporary Parental Level Change feature built into the player" (4.6.4.1 VI4-197).
- The DVD specification does not describe any mechanism to allow the user to play multi-segment parent level branches.

- The DVD2 API computes the minimum level required to play the block and returns this value along with the 'playback stopped' notification. The application can then notify the user of the required parental level that is required to continue playing the content. The user no longer has to guess the required level through trial and error.

- The DVD2 API extends the functionality of the DVD Annex J specification and the first MicroSoft DVD API.
  - The DVD Annex J specification only specifies actions to perform. It does not specify how the application finds out information about the disc or the DVD Navigator's state.
  - New disc and navigation state query functionality:
    - the DVD2 API does not require the application writer to already own a copy of the DVD specification to use it (due to the incomplete description of the data returned by the API). The data returned by the methods to get the textual information, the title attributes, audio attributes and subpicture attributes is documented so that application developers can get the necessary information from the new API and the associated documentation.
    - The DVD2 API allows the application to query the attributes of arbitrary title indices instead of just the current title index.
    - DVD2 API also returns the audio stream's Karaoke information so that intelligent Karaoke applications can be implemented.
    - The DVD2 API also returns the capabilities of the DVD decoder on the system so that the application can present configuration options to the user (like frame stepping in both direction, smooth rewind and fast-forward etc.).
  - New control functionality:
    - The DVD2 API allows the application to play ranges of chapters or ranges of times, to select specific menu buttons (just not relative buttons) and lets the user select buttons using the mouse location.
    - It also supports the getting/setting of bookmark objects (see Bookmark patent) and the ability to query the current unique disc ID (see Unique ID patent)

Command/Operation (?) synchronization mechanism

- the DVD2 API creates associated synchronization command objects. The command object allows the application to synchronize and to learn about the command's status. Each API method is extended with two extra arguments. The general form of a DVD2 API command is:
  - HRESULT IDVDControl2::Command(arguments, DWORD dwFlags, IDvdCmd** ppObj)
    where:
    - IDvdCmd** ppObject is an argument used to return a synchronization COM (Component Object Model) object to the application.

- DWORD dwFlags is the set of flags passed to the method to determine the behavior and usage of the synchronization object. These are a bit-wise union of the available pre-defined flags.

- The synchronization object has the following interface:

```
interface IDvdCmd : IUnknown {
  HRESULT WaitForStart();
  HRESULT WaitForEnd();
}
```

  - The object returned must be released by the application. By returning a pre-incremented COM object, the life of the object can be correctly maintained.

- The flags take the following values:
DVD_CMD_FLAG_SendEvents – events are sent regarding the request's status
DVD_CMD_FLAG_Block – do not continue until the command has been completed
DVD_CMD_FLAG_None – a placeholder indicating no flags

- A special return code is returned by the initial DVD API method, by the IDvdCmd::WaitForStart or IDvdCmd::WaitForEnd or along with the event:
  - VFW_E_DVD_CMD_CANCELLED – the command was pre-empted and is no longer valid

- A sample example of C++ usage of a command object is as follows:

```
IDvdCmd* pObj;
HRESULT hres = IDvdControl2->PlayTitle(15,DVD_CMD_FLAG_None,&pObj );
// don't wait or notify
pObj->Release();
The above call has to be changed a little bit to make it work on an
interface pointer or an interface.
```

- The application can determine the commencement and completion of the command, by either of the following
  - using the command object directly,
  - using no command objects,
  - listening to command related events,
  - using a combination of events and objects to aid in tracking multiple instances of a command. (??)

1) Using objects

- By passing an IDvdCmd pointer to the command, the Navigator will allocate and return a new IDvdCmd object. Calling the interface method IDvdCmd::WaitForStart() will block until the command begins and IDvdCmd::WaitForEnd() waits until the command completes. If the command has been cancelled, then the Navigator will return VFW_E_COMMAND_CANCELLED. After the application is done with the object, it must call Release() to free the COM object. A NULL pointer passed to the DVD API indicates that no command object should be returned to the application and the command execution should continue in the standard asynchronous mode.

2) Not using Objects

- Instead of managing an object, the application can simply specify the DVD_CMD_FLAG_Block flag with a null object pointer. The command will not return until it has either completed or was cancelled. The API will emulate a synchronous behavior. For example:

```
HRESULT hres = IDvdControl2->PlayTitle(uTitle, DVD_CMD_FLAG_Block,
0);
The above call has to be changed a little bit to make it work on an
interface pointer or an interface.
```

is sematically equivalent to:

```
IDvdCmd* pObj;
HRESULT hres = IDvdControl2->PlayTitle( uTitle,
DVD_CMD_FLAG_Block, &pObj);
If( succeeded( hres)) {
      Hres = pObj->WaitToEnd();
      pObj->Release();
}
```

3) Using Events

- Specifying the DVD_CMD_FLAG_SendEvents flag will cause the Navigator to issue the following events:
  - {EC_DVD_CMD_START, LONG_PTR lParam1, HRESULT}
  - {EC_DVD_CMD_END, LONG_PTR lParam1, HRESULT}
- If an application only needs to synchronize one command (or does not differentiate between command instances), no synchronization object is needed and only events are required. A NULL object pointer is passed to the DVD API method and the lParam1 value sent with the event will always be set to 0.

4) Using Events and Objects

- By specifying both objects and the DVD_CMD_FLAG_SendEvents flag, an application can track different commands. The DVD2 API call will return an object that the application can use for later reference.
- When the event notification is sent, the DVD2 API generates a unique identifier (or 'cookie') lParam1 for each event that the application can map back to a DVDcommand(?) object. The cookie approach ensures that applications will not leak memory if they miss an event and allows the DVD Navigator to verify the object.
- The DVD2 API method IDvdInfo2::GetCmdFromEvent(lParam1) maps the cookie into a command object pointer. The application must call Release() on the returned pointer after it has finished processing each of these events.
- When the application is completely finished with the message (after an END event), it must call Release() on the global command pointer that it saved.

Example Blocking/Non-Blocking code

- The following (illustrative?) examples show how synchronization can be accomplished using the IDvdControl2 interface:

- For clarity, some of the examples refer to the following utility function used to map the lParam1 value from EC_DVD_CMD events into an IDvdCmd object:

```
IDvdCmd* GetDvdCmd( LONG_PTR lParam )
{
   IDvdCmd* pCmd;
   pIDvdInfo2->GetCmdFromEvent(lParam, &pCmd);
   return pCmd;
}
```

1) No synchronization (Asynchronous model)

```
HRESULT hres = IDvdControl2->PlayTitle( uTitle, 0, NULL);
   The above call has to be changed a little bit to make it work on an
   interface pointer or an interface.
```

2) Synchronization without events

- An example of the correct way to wait for a command without using events is:

```
IDvdCmd* pObj;
HRESULT hres = IDvdControl2->PlayTitle( uTitle, 0, &pObj);
If( SUCCEEDED( hres)) {
   pObj->WaitToEnd();
   pObj->Release();
}
```

3) Partial synchronization using events

- To synchronize a single event without managing IDvdCmd objects:

```
HRESULT hres = IDvdControl2->PlayTitle( uTitle,
   DVD_CMD_FLAG_SendEvents, NULL);

… We need to put something in here that explains what is lEvent

switch (lEvent)
{
case EC_DVD_CMD_END:
   // IDvdCmd* pObj = GetDvdCmd(lParam1); ← null  ???
   HRESULT hres = HRESULT(lParam2);  ???
   break;
}
```

4) Full synchronization using events

- An example of the correct way to wait for a command using events is:

```
// in global code
IDvdCmd* pGlobalObj = 0 ;
…
// Note: pGlobalObj is assigned by the Navigator BEFORE the event
// is issued; otherwise the event can occur at (*1) before
// pGlobalObj is initialized (?)
HRESULT hres = IDvdControl2->PlayTitle( uTitle,
   DVD_CMD_FLAG_SendEvents, &pGlobalObj );
// (*1)
If( FAILED( hres)) {
   pGlobalObj = NULL;
}
… // some explanation of what is lEvent
switch (lEvent)
{
   case EC_DVD_CMD_END:
      IDvdCmd* pObj = GetDvdCmd( lParam1 );
      HRESULT hres = HRESULT( lParam2 );  ??
      If( NULL != pObj ) {
         If(pGlobalObj == obj ) {  // what does it mean??
            …
            pGlobalObj->Release();
            pGlobalObj = NULL;
         }
         pObj->Release();
      }
      break;
```

5) Full synchronization using events and a separate event loop thread

- An example of the correct way to wait for a command using events is:

```
// in global code
```

```
IDvdCmd* pGlobalObj=0;
CCritSec globalCritSect;
…
{
   CAutoLock(&globalCritSect );  // does CAutoLock take address?
   HRESULT hres = IDvdControl2->PlayTitle( uTitle,
      DVD_CMD_FLAG_SendEvents, &pGlobalObj );
   If( FAILED( hres)) {
      pGlobalObj = NULL;
   }
}
…  // some background of lEvent
switch (lEvent)
{
   case EC_DVD_CMD_COMPLETE:
   case EC_DVD_CMD_CANCEL:
   {
      CautoLock(globalCritSect );

      IDvdCmd* pObj = GetDvdCmd( lParam1 );
      HRESULT hres = HRESULT( lParam2 ); ???
      If( NULL != pObj ) {
         If(pGlobalObj == obj ) {
            …
            pGlobalObj->Release();
            pGlobalObj = NULL;
         }
         pObj->Release();
      }
      break;
   }
```

Playback Interactivity control mechanism

-

  The DVD2 API commands extend the command object mechanism with the following flags:
  - DVD_CMD_FLAG_Flush
  - DVD_CMD_FLAG_StartWhenRendered
  - DVD_CMD_FLAG_EndAfterRendered
- The .._Flush flag indicates that the content should be truncated (like before). The absence of the flag indicates that the current content presentation should end first.
- The ..._..Rendered flags change the semantics of the start and end of each command. By default, the command starts and ends once it has been processed. The new flags indicate that the start and end occur when the results of the change of content have been processed and presented respectively.

Disc communication Mechanism

- The DVD2 API permits applications not only to read the DVD Navigator's general purpose registers (the GPRMs), but also allows them to set the GPRMs using:
  - IDvdInfo2::GetAllGPRMs(WORD pwRegisterArray[16]);
  - IDvdControl2::SetGPRM(ULONG ulIndex, WORD wValue, DWORD dwFlags, IDvdCmd** ppCmd);
- The combined read/write functionality allows DVD applications to 'communicate' with the program on the disc and can implement 'controlled unlocking' or restricted access to the content. The application can use GetAllGPRMs() to read the current state and set a specific register using SetGPRM().
- The SetGPRM() method can also be used to synchronize the application and the DVD Navigator's virtual CPU. The SetGPRM() method is executed only during the periods when the DVD Navigator

is allowed to process user commands (the Presentation and Still phases, 3.3.6.1 VI3-28). Navigation command execution is considered to be atomic. So setting the GPRM is postponed until these phases occur. The application can use the command object to ensure coordination. The disc can loop in a program chain waiting for a GPRM register to be set and can notify the application by performing a Domain change. The command object's event mechanism is serialized with the Domain change event notifications. The application can call SetGPRM() and wait until the command completion event is received, and then wait for a Domain change.

Query Interfaces

- Even though the DVD specification does not suggest any data retrieval methods, both the Microsoft DVD API and DVD2 APIs do provide this capability. The Microsoft DVD API and DVD2 APIs share the following methods:
  - GetAllGPRMs
  - GetAllSPRMs
  - GetAudioLanguage
  - GetCurrentAngle
  - GetCurrentAudio
  - GetCurrentButton
  - GetCurrentDomain
  - GetCurrentLocation
  - GetCurrentSubpicture
  - GetNumberOfChapters
  - GetPlayerParentalLevel
  - GetSubpictureLanguage
  - GetTotalTitleTime
  - GetTitleParentalLevels
  - GetCurrentUOPS
  - GetCurrentVolumeInfo (IDVD2::GetDVDVolumeInfo)
  - GetDVDDirectory (IDVD2::GetRoot1)
- The DVD2 API extends the functionality of the Microsoft DVD API for the attribute query commands by allowing then to work for arbitrary titles and removing the reliance on the DVD specification for:
  - GetAudioAttributes([in] ULONG ulStream, [out] DVD_AudioAttributes *pATR);
  - GetCurrentVideoAttributes([out] DVD_VideoAttributes * pATR);
  - GetVMGAttributes([out] DVD_MenuAttributes * pATR);
  - GetTitleAttributes(ULONG ulTitle, [out] DVD_MenuAttributes * pMenu, [out] DVD_TitleAttributes * pTitle);
  - GetSubpictureAttributes([in] ULONG ulStream, [out] DVD_SubpictureAttributes *pATR);
- It also adds new functionality beyond the DVD API with the methods:
  - Button query
    - GetButtonAtPosition(POINT point, [out] ULONG *puButtonIndex);
    - GetButtonRect(ULONG ulButton,RECT *pRect);
  - Default language settings
    - GetDefaultAudioLanguage(LCID* pLanguage, DVD_AUDIO_LANG_EXT* pAudioExt);
    - GetDefaultMenuLanguage(LCID* pLanguage);
    - GetDefaultSubpictureLanguage(LCID* pLanguage, DVD_SUBPICTURE_LANG_EXT* pSubpictureExtension);
  - DVD text query methods
    - GetDVDTextInfo1?(BYTE *pTextManager, ULONG cbBufSize, ULONG *pcbActualSize);
    - GetDVDTextLanguageInfo(ULONG ulLangIndex, ULONG* pulNumOfStrings, LCID* pwLangCode, DVD_TextCharSet * pbCharacterSet);
    - GetDVDTextNumberOfLanguages(ULONG * pulNumOfLangs);
    - GetDVDTextStringAsNative(ULONG ulLangIndex, ULONG ulStringIndex, BYTE* pbBuffer, ULONG ulMaxBufferSize, ULONG* pulActualSize, enum DVD_TextStringType* pTyp);

- GetDVDTextStringAsUnicode(ULONG ulLangIndex, ULONG ulStringIndex, WCHAR* pchBuffer, ULONG ulMaxBufferSize, ULONG* pActualSize, DVD_TextStringType* pType);
- Miscellaneous
  - GetCmdFromEvent(LONG_PTR dwID, IDvdCmd** ppCmd);
  - GetDecoderCaps(DVD_DECODER_CAPS *pCaps);
  - GetDiscID(LPCWSTR pszwPath, ULONGLONG* pullUniqueID);
  - GetKaraokeAttributes([in] ULONG ulStream, DVD_KaraokeAttributes *pATR);
  - GetMenuLanguages(LCID *pLang, ULONG uMaxLang, ULONG *puActualLang);
  - IsAudioStreamEnabled(ULONG ulStreamNum, BOOL *pbEnabled);
  - IsSubpictureStreamEnabled(ULONG ulStreamNum, BOOL *pbEnabled);

Simplified Naming Scheme

- The DVD2 API contains a simplified naming scheme for the potential user operations suggested in the DVD specification Annex J. The names now suggest their usage instead of an abstract label. Below is a summary of the names:

Annex J	DVD1	DVD2
Angle_Change	AngleChange	SelectAngle
Audio_Stream_Change	AudioStreamChange	SelectAudioStream
Backward_Scan	BackwardScan	PlayBackwards
Forward_Scan	ForwardScan	PlayForwards
PTT_Search	ChapterSearch	PlayChapter
PTT_Play	ChapterPlay	PlayChapterInTitle
Button_Activate	ButtonActivate	ActivateButton
-	ChapterPlayAutoStop	PlayChaptersAutoStop
Pause_Off	PauseOff	Pause(false)
Pause_On	PauseOn	Pause(true)
Karaoke_Audio_Presentation_Mode_Change	KaraokeAudioPresentationModeChange	SelectKaraokeAudioPresentationMode
Menu_Call	MenuCall	ShowMenu
Still_Off	StillOff	StillOff
Stop	Stop	Stop
-	StopForResume	-
Resume	Resume	Resume
-	SetRoot	SetDVDDirectory
Button_Select_And_Activate	ButtonSelectAndActivate	SelectAndActivateButton
NextPG_Search	NextPGSearch	PlayNextChapter
PrevPG_Search	PrevPGSearch	PlayPrevChapter
Title_Play	TitlePlay	PlayTitle
Time_Search	TimeSearch	PlayAtTime
Time_Play	TimePlay	PlayAtTimeInTitle
-	MouseSelect	SelectAtPosition
-	MouseActivate	ActivateAtPosition
Video_Presentation_Mode_Change	VideoModePreferrence	SelectVideoModePreference
Sub-picture_Stream_Change	SubpictureStreamChange	SelectSubpictureStream
		SetSubpictureState
TopPG_Search	TopPGSearch	ReplayChapter
Left_Button_Select	LeftButtonSelect	SelectRelativeButton(Left)
Lower_Button_Select	LowerButtonSelect	SelectRelativeButton(Lower)

Upper_Button_Select	UpperButtonSelect	SelectRelativeButton(Upper)
Right_Button_Select	RightButtonSelect	SelectRelativeButton(Right)
Menu_Language_Select	MenuLanguageSelect	SelectDefaultMenuLanguage
Parental_Country_Select	ParentalCountrySelect	SelectParentalCountry
Parental_Level_Select	ParentalLevelSelect	SelectParentalLevel
GoUp	GoUp	ReturnFromSubmenu

New Control Interfaces

1) Period Playback Interface

- In addition to playing ranges of chapters, the DVD2 API allows the playing of time periods using:
  - PlayPeriodInTitleAutoStop(ULONG ulTitle, DVD_HMSF_TIMECODE* pStartTime, DVD_HMSF_TIMECODE* pEndTime, DWORD dwFlags, IDvdCmd** ppCmd)
- With this method, applications (such as video editing programs and games) can accurately play back arbitrary portions of the content.
- Combined with the command object mechanism, any application like slideshow presentation, video games interludes, or kiosks can be implemented using a single DVD2 API command.

2) Default language Interfaces

- SelectDefaultAudioLanguage(LCID Language, DVD_AUDIO_LANG_EXT audioExtension)
- SelectDefaultSubpictureLanguage(LCID Language, DVD_SUBPICTURE_LANG_EXT subpictureExtension)

These methods allow applications (from user) to set the default language choices for DVD playback.

3) Button index selection

- Applications can now automate menu navigation through the method
  SelectButton(ULONG ulButton)

4) Bookmarking APIs

- Applications can save and restore the entire DVD state (see bookmark patent)
  - GetState(IDvdState **pStateData)
  - SetState(IDvdState* pState, DWORD dwFlags, [out] IDvdCmd** ppCmd)

5) Other

- AcceptParentalLevelChange(BOOL bAccept) – see "Minimum parental level branching"
- SetGPRM(ULONG ulIndex, WORD wValue, DWORD dwFlags, [out] IDvdCmd** ppCmd) – See "Disc communication Mechanism"
- SetOption(DVD_OPTION_FLAG flag, BOOL bEnable) – extendible option setting mechanism

Mechanism for coordinating minimum parental level branching

- According to the DVD specification (section 4.6.4.1 pVI4-197), when the DVD Navigator encounters a 'SetTmpPML' (set temporary parental management level; see DVD Spec) command, it should request permission from the application ("call the Temporary Parental level Change feature built into the player") to temporarily raise the current level. If the parental level change is allowed, the Navigator raises the parental level and branches to the restricted piece of content. Otherwise, it continues with the next command.

- Under the DVD2 API, the following sequence occurs:
  - The application notifies the API of the availability of the parental level change feature by calling the IDVDControl2::SetOption(DVD_NotifyParentalLevelChange, TRUE).
  - When the DVD Navigator encounters a SetTmpPML instruction, it sends a PARENTAL_LEVEL_TOO_LOW event to the application. The application is expected to put up some user interface to let the user bump up the parental level.
  - The DVD Navigator blocks until the application responds by calling IDVDControl2::AcceptParentalLevelChange() with TRUE or FALSE, and then proceeds accordingly.

Mechanism for aiding playback of multi-segment parental level branches

- The DVD specification (Section 4.1.4.1 VI4-22) describes a scheme for selecting different program chains (usually different segments of content (?)) based on the current parental level. For each title, the PTL_MAI table maps the current parental level into a 16-bit mask.
- During playback, the DVD Navigator uses the PTL_MAI table to obtain the current parental bit mask. The parental bit mask is used when the Navigator encounters a parental block (a collection of program chains in which each program chain has an exclusive parental bit mask). The Navigator searches each PTL_ID_FLD in the VTS_PGCI_SRP (Section 4.2.3 VI4-62) for a program chain with a bit mask that shares common bits with the current parental bit mask.
- If no program chain partially matches the current bit mask, the old implementation of Microsoft DVD Navigator halts the playback and sends a DVD_ERROR_LowParentalLevel event to the application.
- To help the user, the DVD2 API uses the following algorithm to compute the minimum required parental level that would let the user continue:
  - Initialize $PTL\_MASK = 0$ (the possible allowed parental levels)
  - For each program chain index i in the VTS_PGCI_SRP
    - If $VTS\_PGCI\_SRP[i].BlockType = 1$ (in a parental block)
      - $PTL\_MASK = PTL\_MASK \,Y\, VTS\_PGCI\_SRP[i].PTL\_ID\_FLD$
  - If $PTL\_MASK = 0$ then no parental level is present, so any level will work
  - Else for each parental level index i in the PTL_MAI
    - Let $PTL\_LVLI = PTL\_MAI[8-i]$
    - If $PTL\_LVLI[title\_index]\,I\; PTL\_MAI[8-i] \neq 0$ (note: $title\_index = 0$ in VMGM)
      - Return i
- The index i is returned along with the DVD_ERROR_LowParentalLevel event. The application can use the index to suggest a possible parental level setting to the user.

Frequently Asked Questions
Windows Millennium Edition

Revision - 1.6 (February 12th 2002)

Windows Millennium Edition

Q: What is Windows Millennium Edition?
A: Windows Millennium Edition was the last release of the Windows 9* operating system. It is designed solely for the home user.

Q: What is the build number for the final version of Windows Me?
A: 3000.2

Q: When can I buy Windows Me?
A: You can buy Windows Me from September 14th. MSDN users will receive their copies on CD in July while new PC buyers can expect to see Windows Me on PC's late July.

Q: How much will Windows Me cost?
A: $209 for the full version, $109 for the upgrade.

Q: What will be new in Windows Millennium Edition?
A: Not a great deal is new in Windows "Millennium Edition" despite the hype that is sure to come out around its release date. Here are the basics from the Press Release Microsoft sent out.

- "Digital Media and Entertainment" Digital media is becoming increasingly popular, as illustrated by the exponential growth in areas like music on the Web and digital photography. The Consumer Windows Division will focus on enabling users to take advantage of all this new content, making it easy to access, play/view and store as well as providing an enhanced PC gaming experience.

Windows 2000
Windows Me
Windows Server 2002
Windows "Whistler" XP
Windows CE
Internet Explorer 6
Internet Explorer 5
Xbox
Xbox 360
DirectX
DVD's
TopTechTips
Registry Tips
Windows 95/98
Windows 2000
Internet Explorer 5
Program Tips
Easter Eggs
Hardware
DVD
ActiveDVD
DVD News
DVD Forum
Glossary
Tips
Articles
Reviews
News Archive
Links
Drivers
Latest Reviews
Games
Halo (PC)
Xbox
Mafia
Rainbow 6: 3
Applications
JBuilder X Enterprise Edition
Office 2003
Hardware
AMD Athlon 64 4000+
Microsoft Fingerprint Reader
Latest Interviews
Steve Ballmer
Jim Allchin
Site News/Info
About This Site
Affiliates
ANet Forums
Contact Us
Default Home Page
Link To Us
Links

- "Online Experience" Providing consumers a premier home online experience is a primary goal for the Consumer Windows Division. This means ensuring consumers can easily connect to the Web, locate desired content and determine which content is right for their family.

- "Home Networking" With more than 15 million households now owning two computers and the cost of new PCs continuing to fall, as well as the many intelligent hardware devices being created, networking at home is becoming a reality for more people. The Consumer Windows Division will work to simplify the process of connecting multiple computers in the home, enabling them to share information and an Internet connection, and provide the infrastructure for connecting different intelligent devices to the PC.

- "It Just Works" The Consumer Windows Division is committed to providing consumers with a solution that 'just works,' from the moment a user starts their PC and throughout their daily computing experience. This promise will be delivered upon by the advancement of the PC's self-healing functionality, in addition to providing a simpler set-up and a great out-of-the-box experience for new computer users.

There are a number of other new features such as the following:

- **Universal Plug and Play:** Universal Plug and Play (UPnP) defines an architectural framework for creating self-configuring, self-describing devices and services. It enables seamless connectivity and communication between Windows and intelligent appliances in the home, office and everywhere in between.

Q: When did testing begin on Windows Me?
A: Testing first began back in September 1999 when we first broke the story.

Q: What can we expect to see in Windows Millennium Edition that will make us upgrade from Windows 98?
A: The major plus points are a much faster booting time (30 seconds for some new users) and far great stability.

Q: If I am a games player, should I go for Windows 2000 or wait for Millennium Edition?
A: Defiantly wait for Windows Millennium Edition for the simple reason that it is designed for the consumer thus is more adept at games playing.

Q: Will my games run faster and smoother on Windows Me compared to Windows 98?
A: No, even though Windows Me ships with a newer version of DirectX 7, there is little speed increase in any game we have tried out, the main difference is stability across all fronts.

Q: I hear that MS-DOS has gone, how will I run some older programs like Scandisk etc?
A: MS-DOS isn't gone, its just harder to get too. Just create a Windows Start Up disk and you can easily get to MS-Dos. Programs like Scandisk no longer need to run in Dos if there is a bad shutdown, Microsoft has improved the program to run when Windows restarts.

Q: I have been told that I can no longer update my Bios due to the lack of MS-DOS in Windows Me, is this true?
A: No. Creating a boot disk to run a bios update is easily done. Read our tip for details on how to create one.

Q: Is the version of Internet Explorer 5.5 in Windows Me older than the final released to the public?

Member Pages
News Archive
Site Search
Awards

A: It seems so. Installing Windows Me over Internet Explorer 5.5 final keeps the newer version that you downloaded.

Q: Does Windows Millennium Edition support dual processors like Windows 2000 does?
A: No - Windows Me is still based on the Windows 9* core and the first consumer edition of Windows that will support Dual Processors will be Windows Whistler (Windows 2001)

Q: I have just purchased Windows Me - have any updates been released for it yet?
A: Yes! Many parts of Windows Me are already well out of date including Internet Explorer 5.5, Windows Media Player 7 and some security updates so pop along to http://windowsupdate.microsoft.com

Q: Do you have a review of Windows Millennium Edition?
A: Yes we have posted our review of Windows Me on our site.

Q: Where can I find more information?
A: Visit our Windows Me Section or Microsoft's Windows Me site.

Q: I have Windows Me - is it advisable to upgrade to Windows XP?
A: Yes without a doubt we wholly recommend upgrading to Windows XP and as you are a Windows Me user, we recommend that you purchase Windows XP Home Edition for your PC.

Windows Millennium Edition Timeline

First News Of "Millennium" Broken: Summer 1999 Exclusively On ActiveWin
Developer Preview 1: Released July 23 1999
Microsoft Officially Announces "Millennium": July 26th 1999
Beta 1: Released September 23 1999
Beta 2: Released November 23 1999
Beta 2 Refresh: Released January 20 2000
Officially Named Windows Millennium Edition (Windows Me): January 31st 2000
Beta 3: Completed April 7th 2000 - Released 11th April
RC 0: May 9th 2000
RC 1: May 17th 2000
RTM: June 18th 2000
Final release: In stores September 14th. On PC's Late July

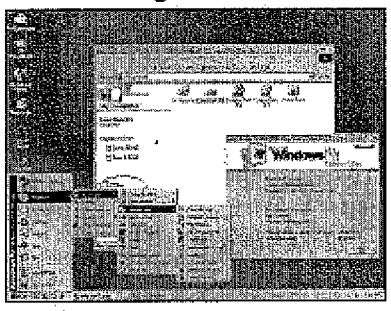
On The Internet

Windows Me
The Official website for Windows Millennium Edition

Windows Me Review
Our review of the gold release

Return to the FAQ index

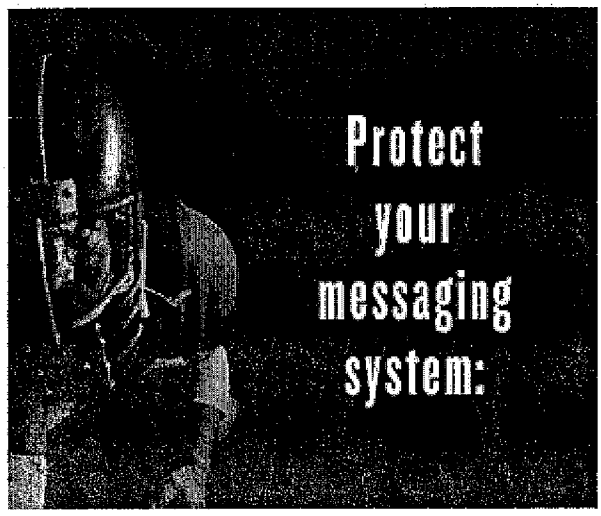
Introducing Windows Millennium Edition ("Windows Me") Beta 3



Windows Millennium Edition ("Windows Me") Beta 3 looks and feels like Windows 2000 but offers a host of new consumer-oriented features.

For years, Microsoft had plotted the demise of the old DOS/Windows family of products with a set of point releases that would follow Windows 95. The first of these--known as OEM Service Releases (OSR)--followed the original release of Windows 95 rather closely, adding few new features other than the now infamous Internet Explorer browser integration. And of course, in 1998, Microsoft released Windows 98, the first Windows 95 follow-up to be sold at retail. And in 1999, Windows 98 Second Edition arrived. Each of these sequels to Windows 95 fixed bugs and added new features, but none were dramatic departures from what had come before.

The reason for this lay in Microsoft's enterprise operating system, Windows NT, which was built from the ground up in the early 1990's to be reliable and secure, two features that have continually eluded the company's consumer offerings. Early versions of Windows NT, however, were to large and slow to run acceptably on consumer grade hardware, and software and hardware compatibility issued heaped more frustration on anyone that wanted to upgrade to Microsoft's high-level OS. But Microsoft knew that it would be only a matter of time before the typical consumer machine could run Windows NT, and it had been working to make NT more compatible with its DOS-based siblings since the release of Windows 95.



The original plan, perhaps, was a bit too grandiose: After the release of Windows NT 4.0, the company planned to phase out the 9x line with the release of Windows 98. Windows NT 5.0, which was then expected to ship in 1999, would eventually produce a consumer grade edition that would be suitable for home users. However lofty the goals, however, Microsoft was unable to capitalize on these plans. As Windows NT 5.0 slipped from 1999 into 2000, Microsoft issued a second release of Windows 98 and pushed back plans for a consumer version of Windows NT 5.0, which was renamed to Windows 2000. The company promised that a future version of Windows 2000--now code-named "Whistler"--would include a consumer version. But that left a bit of a hole: The company had been releasing yearly updates to its consumer OS since 1995, and no new release was now ready to go for 2000.

Enter Windows Millennium Edition ("Windows Me"), the final version of Windows 98 and the last release of the old DOS/Windows codebase. Announced in July 1999, Windows Me was initially known only by its code-name, "Windows Millennium." Internet rumor sites excitedly announced that Millennium would include a new HTML-based user interface featuring tight integration with consumer-oriented features such as MP3 audio, video, and the Internet. But the initial reports were pretty far off base: Though the

company was indeed working on HTML user interfaces, they weren't far enough along to be included in Millennium as the main user interface. On the other hand, a number of HTML-based "Activity Centers"--originally slated to appear in the consumer version of NT 5.0 (then called "Neptune")--will make it into Millennium: These include the online help system, called Help & Support, and a new system recovery feature called System Restore.

The Millennium beta began in fall 1999 with a rather uninspired developer's preview release that didn't add much at all to a stock Windows 98 Second Edition install. But unlike the rumored HTML user interface, there was one planned feature that made it into even the first builds of Millennium, the removal of Real Mode DOS. Though most average users don't understand the interaction of DOS and Windows in Windows 3.1 and 9x releases, Windows has long been a standalone operating system that didn't require the presence of a legacy DOS system to operate correctly. Beginning with Windows 3.1 in 1992, Windows used DOS solely as a boot loader, and it bypasses DOS completely for memory and file system access. On these modern Windows operating systems, DOS is actually run as a virtual machine inside of Windows, not the other way around as most people think. However, Windows 3.1 and all 9x releases through Windows 98 also provide a legacy Real Mode DOS, which the system uses to boot. When you boot into a Windows 9x command line, or "restart in MS-DOS mode", what you're really doing is loading Real Mode DOS. Millennium removes this support for reliability purposes; it also causes the system to boot and shut down much more quickly than before because the legacy DOS mode can be skipped. But though this means that Millennium cannot boot to a command line (without a boot floppy), most DOS programs should continue to run under Windows as before. It's the ultimate "best of both worlds" situation.

After the initial developer preview release, early beta builds added a few new features, though none of them were particularly compelling. When Millennium Beta 1 was released on September 24, 1999, Microsoft even declined to supply review copies to the press, because it felt that the OS didn't yet accurately portray what the final product would look like. On November 24, 1999, Microsoft released Windows Millennium Beta 2 (please read my introduction and exhaustive review), which still largely resembled Windows 98 SE, though some new features such as primitive HTML-based Help and System Restore, set it apart somewhat. Still, at that point in time, Millennium seemed like an uninspired release designed solely to fill the gap between Windows 98 SE and Whistler.

Since that time, however, Millennium has improved dramatically. Unfortunately, for the first time, it also fell behind schedule. On January 21, 2000, Microsoft released what the company called a "Beta 2 Refresh," at a time when Beta 3 was originally expected. And on February 1, 2000, Microsoft announced that Millennium would be marketed under the name Windows Millennium Edition ("Windows Me"). The company tells me that several other names were considered (remember, "Windows 2000" was taken by the renamed NT 5.0), including Windows Home, Windows Family, and Windows Personal, but consumer testing put "Windows Me" over the top. Millennium--Windows Me--now sported the Windows 2000 shell and color scheme, finally dropping the dark gray and teal scheme that had been used for the past five years. And though some HTML-like improvements were scaled back (such as AutoUpdate), the System Restore and Help & Support tools, especially, started to take on a newfound professionalism.

Windows Me Beta 3 debuted two months late, a delay that was largely due to a complete rewrite of the old Windows 98 TCP/IP stack and some incompatibilities that this caused with certain network cards. It includes Internet Explorer 5.5 Beta 3 and the first beta of Microsoft Media Player 7, a completely new multimedia player that more closely competes with Winamp and RealPlayer than previous versions. And with the addition of these features, and new reliability features such as System Restore and the System File Protection (SFP) functionality from Windows 2000, Windows Me is a compelling upgrade for all users of Windows 9x.

In my upcoming review of Windows Millennium Edition ("Windows Me") Beta 3, I will examine the latest release of Microsoft's upcoming consumer operating system, holding it up against the company's own goals for the OS. Microsoft says that Windows Me will focus on digital media, the online experience, PC health, and home networking, the four areas that are of most interest to potential customers. Also,

because Windows Me is designed to easily and seamlessly upgrade legacy Windows 9x systems, I take a look at the upgrade picture in addition to simply installing Windows Me Beta 3 on a fresh system with no previous OS installation.

Coming soon: My exhaustive review of Windows Millennium Edition ("Windows Me") Beta 3!